EXPERIMENT NINE

Experiment No. 09

Aim

To implement the N-Queen problem using backtracking.

Prerequisite

Basic understanding of game theory and recursion.

Learning Outcome

After completing this experiment, you will be able to understand complex problem-solving techniques and implement backtracking algorithms to solve constraint satisfaction problems.

Theory

The N-Queen problem is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. This means that no two queens can be in the same row, column, or diagonal. Backtracking is used to systematically search for a solution by placing queens one by one in different columns and backtracking if a conflict occurs.

Task

Implement the N-Queen problem using Python and the backtracking algorithm. Display all possible solutions for a given value of N.

```
# N-Queen Problem using Backtracking

def print_solution(board):
    N = len(board)
    for i in range(N):
        for j in range(N):
            print('Q' if board[i][j] else '.', end=' ')
        print()
    print()

def is_safe(board, row, col):
    N = len(board)

    # Check this row on left side
    for i in range(col):
        if board[row][i]:
```

```
            return False

    # Check upper diagonal on left side
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j]:
            return False

    # Check lower diagonal on left side
    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j]:
            return False

    return True

def solve_n_queens_util(board, col):
    N = len(board)
    if col >= N:
        print_solution(board)
        return True

    res = False
    for i in range(N):
        if is_safe(board, i, col):
            board[i][col] = True
            res = solve_n_queens_util(board, col + 1) or res
            board[i][col] = False  # BACKTRACK

    return res

def solve_n_queens(N):
    board = [[False] * N for _ in range(N)]
    if not solve_n_queens_util(board, 0):
        print("No solution exists")
        return
    return

# Example: Change N here to solve for different sizes
N = 4
solve_n_queens(N)
```

```
. . Q .
Q . . .
. . . Q
. Q . .

. Q . .
. . . Q
Q . . .
. . Q .
```